Software Construction, Techniques and Tools

OBJECTIVE

By the end of this course, with appropriate use of software students will be able to create a functional setup that they will install in the computer, conceive, analyse and design projects. Software we are going to use include

* Java (jdk)
* Netbeans
* Launch 4J
* Inno Setup
* Xampp

CH1: Generalities on software construction

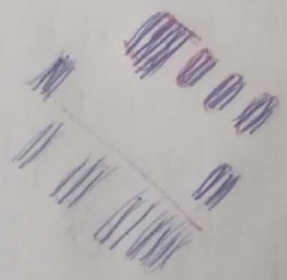
CH2: Software Requirements

CH3: Software Development life Cycle

~~CH4: Creation of an application software.~~

CH 4: Software Development life Cycle Models

CH 5: Creation of an application software



CH1: Generalities on Software Construction

1.1 Def: A software can be defined as a collection of executable programming codes associated with libraries and documentations.

To design a software can be seen as using developing products, using well known and associated scientific principles, methodologies and procedures.

It is the possible to define software design or software construction as an engineering branch associated with development of software products, using well defined and appropriate principles, methods and procedures.

Definitely, software construction is a branch in computer science using well defined engineering concepts to produce efficient, durable, scalable, in-budget, on-time software product.

1.2 Characteristics of a Good Software.

Technically, a good software can be judged by what it offers and how it can be as user friendly as possible. It should satisfy the following criteria.

1.2.1 Operational: The cost should be acceptable for the designer and the user. It should have a high usability, so that each user could use it without a heavy and long training. The efficiency and accuracy should have a high-level (same conditions should produce same results). The software should have high level of security and safety.

Transitional: This means that software should be platform independent.

1.2.5 Maintenance: The software should demonstrate its ability to be adapted to the futur, maintain itself in the environment

1.3 Software Metrics and Measures.

They can be understood as a process of quantifying and symbolising various attributes and aspects of software. Some software metrics are; ~~size metrics~~

1.3.1 Size Metrics: They are the number of lines of codes (LOC) and is usually calculated in thousands of delivered source code lines, denoted (KLOC).

1.3.2 Function Point Count: It defines the size of functional aspects of the software.

1.3.3 Complexity Metrics: It characterises the number of independent paths of the program, or modules.

1.3.4 Quality Metrics: This defines the quality of the product, it is the number of defects, found in development process and the number of defects reported by the clients after the product is installed.

1.3.5 Process Metrics: These are the phases of SDLC.

1.3.6 Resource Metrics: It takes into consideration all the efforts, time and various resources used.

1.4 Software User Interface Design.

User interface is the front end application view, through which users interact in order to use ther software. It can be graphical, textbased, audio-visual based, electro-magnetic wave based, ... The software becomes more popular if it's user interface is

* attractive
* simple to use
* responsive in short-time
* Clear to understand
* Consistent

User interface is divided mostly into 2 categories; command line interface (CLI) and graphical user interface (GUI).

1041 Command Line Interface (CLI). CLI provides a command promt, but the user needs to remember the syntax and command before using. There are methods like macro scripts that make it easy for the user to operate. A textbased command line interface can have the following elements.

* Command prompt
* Cursor
* Command

1042 Graphical User Interface (GUI). This interfaces are made of organised and arranged collection of graphic components, which provide an easy way to work with the system.

Elements are

* window
* Child window
* Menu
* Icons
* Tabs
* Cursor
* Dialogue boxs
* textboxes
* combo boxs
* Drop down list
* Radio buttons
* Check boxs

CH2: Software Requirements.

2.1 Introduction.

Software requirements are the description of features and functionalities of the target system. Requirements convey the expectations of users from the software product.

2.2. Functional Requirements.

It defines the functions within the software system. They are related to the functional aspects of the software.

2.3. Non-functional requirements.

They can be

* Security
* Storage
* Configuration
* Performance
* Cost
* Flexibility
* Disaster recovery
* Accessibility

2.4. User-Interface Requirements

A software is widely accepted if it is

* Easy to operate
* Quick in a response
* Effectively handling operational errors
* Providing simple consistent user interface.
* Have a good presentation
* Easy navigation
* Provide help information
* Strategical use of colors and textures
* Responsive and interactive.

2.5 Requirement engineering Process.

The process to gather the software requirement from client is known as requirement engineering. It is also intergrates analysis and documentations.

2.5.1
2.6 Feasibility study.

When the client approaches the organisation forgetting the desired product developed, he comes up with a number of ideas about what, all functions the software must perform and which features are expected from the software. Referencing to this information, the computer analyst does a detailed study about whether the desired system and it's functionality are feasible to develop. Analysts should take into consideration all available materials and interactions.

2.5.2
2.7 Requirement Gathering.

If the feasibility report is positive and we are sure to develop the project, the next step starts with gathering requirements from the user. Analyst and engineers communicate with the client and end users, to identify their ideas on what the software should provide, and which features they want the software to include.

2.5.3 Software requirement specification

It is a document, created by system analyst after all requirements are collected from various stakeholders.

software requirement validation.

We can use the following conditions

* Can the software be practically implemented?
* Are there any ambiguities?
* Is the software complete?
* Can we demonstrate to people?

2.6 Requirement Elicitation Techniques.

It is the process to find out the requirements for the software system, by communicating with clients, end users, system users and others who have a stake in the software system ~~development~~ development.

2.6.1 Interview:

These are strong mediums to collect requirements. We may conduct several types of interviews such as

* Brainstorming with oral interviews, written interviews one-to-one interviews or group interviews.
* Closed interviews
* Open interviews, here information to gather is not decided in advance.

2.6.2 Questionnaires

It is a document with a predefined set of objective questions and respective options to check

2.6.3 Surveys.

It is just a query about the expectations and requirements for the new coming system.

CH3: System Development Life Cycle (SDLC)

3.1 Introduction.

The realisation of any computer based information system, requires following certain steps and rules in order to fullfill the adopted requirements of the future system to be built. SDLC, describes the d/f stages in the realisation of the system, their organisation, and their succession in time. This chapter is aimed at detailly explaining each phase of the process.

3.2 System Life cycle.

In general, system life cycle is a period of time that ranges from the begining to the end of system process. It begins from proposition or decision to develope a new system and ends at the moment when the system runs out of service. The precise decision of ~~the~~ modeling a system life cycle, details all the activities to be carried out and the time at which it has to be carried out.

3.3 System Development Life Cycle.

The path of the system life cycle, strictly devoted to the implementation to start the system or software development. The SDLC, starts with the decision to develope a new software and ends or finishes with the deliveny of the software and it's installation. The lines below describe the various activities that would be performed in the SDLC and give their order of precedence.

expression of needs and feasibility study.

It is the 1st step in the SDLC. It begins at the particular moment when the idea of building a new system is conceived, or upgrading or enhancing an existing system to satisfy specific needs. These needs can either be expressed by clients with the aim of setting in place the new system or modifying an existing one or by some users, already working on a pre-established system. The expression of needs should clearly express what is expected from the future system, with abstraction of how it should be doing it. It describes the functionalities of the system and also how to use it. This 1st part is therefore, the expression of an idea and focusses on 2 main points.

* What the system will add to the users.
* How will the system react in from of the users.

The expression of needs is elaborated through a dialogue between the clients or users and experts, who have the aptitude to judge if these aspirations are technically realisable within the available budgets allocated and the time allowed. It is usually materialised by a final document which discloses the agreement of both parts.

3.2. Specification of objectives.

The expression of needs is usually not very precise and enough to serve the base of implementation of a new system. It is very important to put in place a step by step elaboration of specification of the

future system. This stage will serve as the foundation for the future system designers. The aim of this stage is to bracket specifically, the new system and describe the different ways how the futur users will use the application. Specification are elaborated by experts of the application domain (clients or future users) in collaboration with those incharge of the ~~deliverable~~ realisation of the system. It is finalised by the deliverable, easy to understand by users and also easy to realise by the technicians. It is advisable to intergrate into the final document diagrams, models and screen sequences to facilitate communication with the future users.

3.3.3 Analysis

The aim of this 3rd stage, is to better understand problems, related to the subject matter, that have been stated, it should be completely free for any technical consideration and also free of any aspect of realisation, such as programing language, the DBMS, material and technical configuration etc. It should rather help both the future users and analysts to validate and complete the system study. The objective is to determine the elements that intervene in the system, their structure, as well as their relationships in order to give a clear, concise and strict definition of the future system. The analysis should always be correct even if it is incomplete.

The analysis agree on "what the system must do" before it agrees on "how to do it". It also focuses on three main axes;

functional axis.

It describes the know-how or ability of the future system.

2. Structural axis or static axis.

It describes the structural representation of the system

3. Dynamic axis

It describes the various states of the system and various events or messages it recieves.

3.3.4. Design.

The designing stage, relies on the document produced at the analysis stage and ~~defines~~ refines them, taking into account the technical environment in which the system operates. It is aimed at determining how to solve the problems studied by the analysis stage and propose solutions for the implementation and the realisation. It therefore requires performing judicious choices on the technical architecture, the performance and optimisation programming strategies and data persistency. The final document at this stage is realised by computer experts and specialists of the programming language used in collaboration with DBMS experts. It should also involve experts of the adopted technology.

3.3.5 Implementation.

It is the stage at which the different structures and various algorithms, previously defined during the design stage, are translated into a particular programming language or a specified database.

It is often seen as the most predominant step of system life cycle and is done at the detriment of the other steps, which are usually bungled due to time constraint. Nevertheless, it is often more expensive to correct a mistake related to the desing in the implementation phase than to correct it in the analysis or design stage. The use of many code generation tools from design models has considerably reduced the implementation stage. Indeed, they exist many available cost free tools, that help programmers easily generate instructions code for an application. This reduces significantly the time allocated for the implementation.

3.3.6 Tests and integration.

The main aim of test consist of verifying if the system has been built properly without any design of programming errors, in order to ensure the technical qualities of the realised system. It is concerned with controlling the workdone by the programmers and raises all the design and implementation defects in order to guarantee the robustness and coherence of the system. It is very important and obvious that those performing the test, should be biased. In other words, the system should not be tested by those who designed and analysed it. The testing stage usually generate 3 types of consequences.

duration and resources allocated to the tests are underestimated in the planning of the project.

2. Errors detected are usually not well perceived by designers and programmers.

3. Errors have important repercusions and side effects because they are dilected very late in the system development life cycle.

3.3.7 Validation.

The client or customer is the center of this stage. Validation can be considered at many levels, depending on the technical competencies of the customers. Validation focuses on the satisfaction of the client. It ensures that the system built is not only robust and coherent, but is also in accordance with the client satisfaction and needs. Validation is not done by those who build the system, but rather by those who demanded and intend to use it. It should also be well planed in terms of time and resources.

3.3.8 Maintenance and evolution.

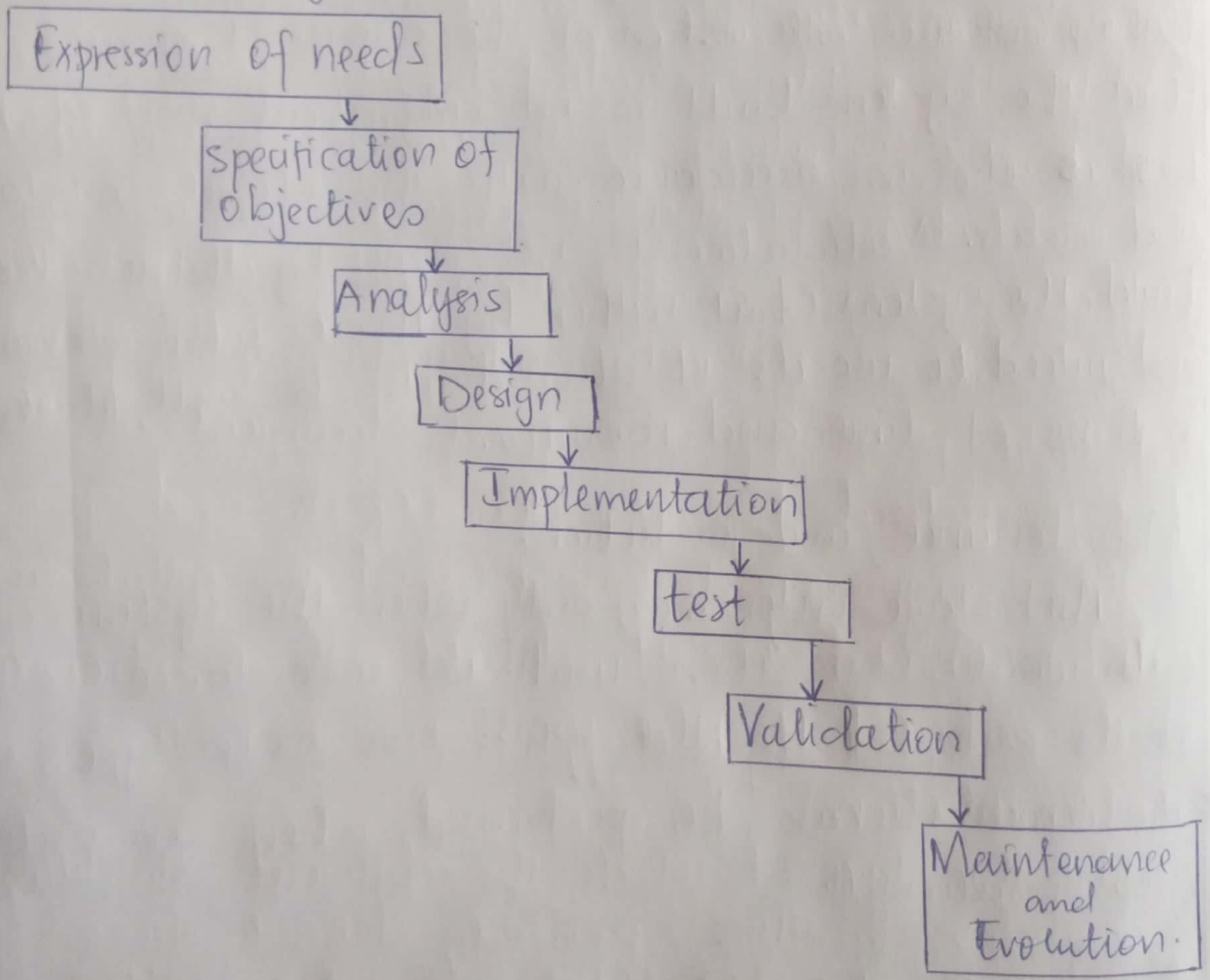
This stage intervenes only when the system goes operational. It is then used in accordance to the needs expressed by the users. Two main types of modifications can be performed, when an application goes operational.

1) Maintenance: It consist of carrying out minor but necessary corrections, in order to aminurate the system robustness and perfomance and also its conviviality.

ii) Evolution: It consist on one hand of adapting the system to meet and respond to modification, such as additional functionalities as they arise. On the other hand, to carry out modification in the internal structure of the system, to correct design errors.

These two operations, constitute two major steps because they are usually the longest in the SDLC and therefore more expensive. They are also very critical because the probability to be carried out depends strongly on the realisation of the system.

In Summary,



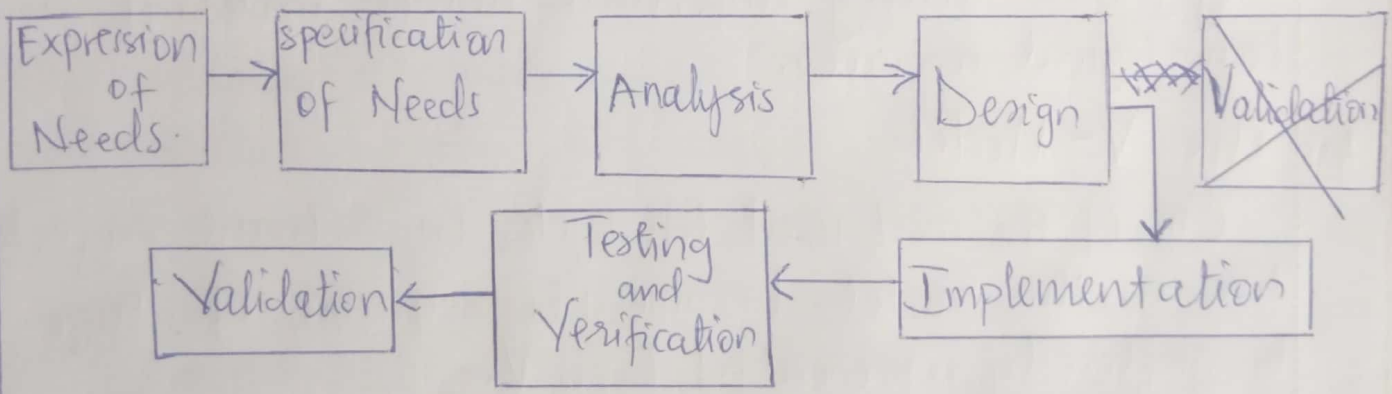
CH4: System Development Life Cycle Models

4.1 Introduction.

The SDLC aims at explaining the sequencing of the different stages explained in the previous chapter and the manner in which the co-operate. This chapter shall focus on presenting some standard life cycle (SDLC) models used in the developement of computer applications.

4.2 Linear or Sequential life Cycle

The life cycle of most software are considered as linear or sequential because the software is often completely specified, completely analysed, completely designed, completely implemented and completely tested and validated. The life-cycle here consist of different stages condition for the begining of another stage.



The Linear Life-Cycle Model.

This type of sequential process presents many inconviaiences that could be very harmful to the running and control of a project.

* There is a breaking point between each stage and feedback effect is very delicate. It makes difficult the commitance of different stages with the risk of

reducing terms productivity and increasing project duration.

* Development of project interfaces and project functionality arises very late in the progress of the project - first auditable results by the clients are awaited with impatience. Potential risk domains are not discovered in time, which does not permit to anticipate or put in place corrective actions within acceptable date lines.

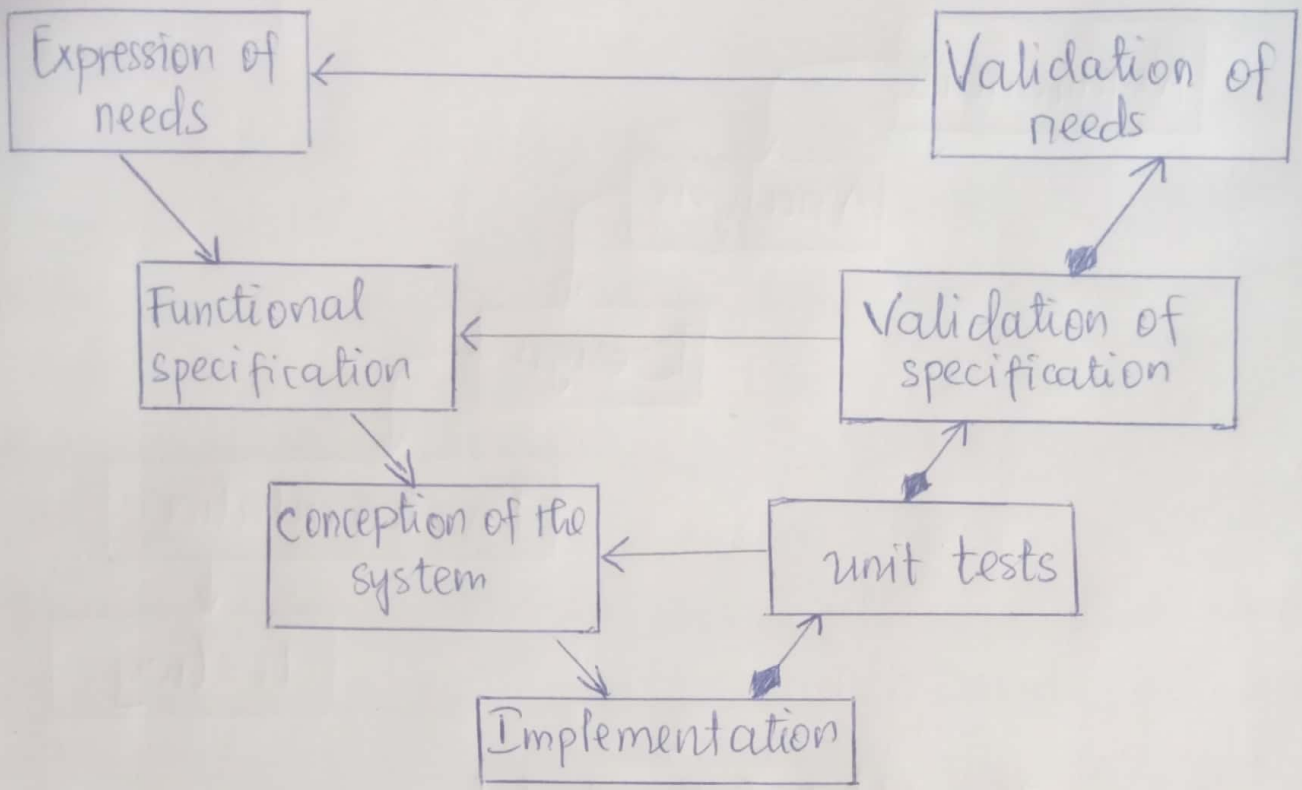
* Validation is carried out late because it intervains at the end of the project and consequently analysis, design and implementation errors become very expensive and dateline exceeded.

Linear or sequential models, make the planning hardly managable and less flexible. This management difficulties are more stressed on very big projects, that sprayout on many years, involving important human material and technological resources.

4.3 The V-Model.

One of the most used life cycles in software projects, lives on this model, because the sequence of stages can be graphically represented by a V.

This model relies on the sequence of the autonomous stages, setting a correspondent between the moments when a need is expressed and the moment when it can be approved and validated. In this model, stages related to the analysis and design are set in place in a descending approach, while those related in the verification approach are realised in the ascending approach.



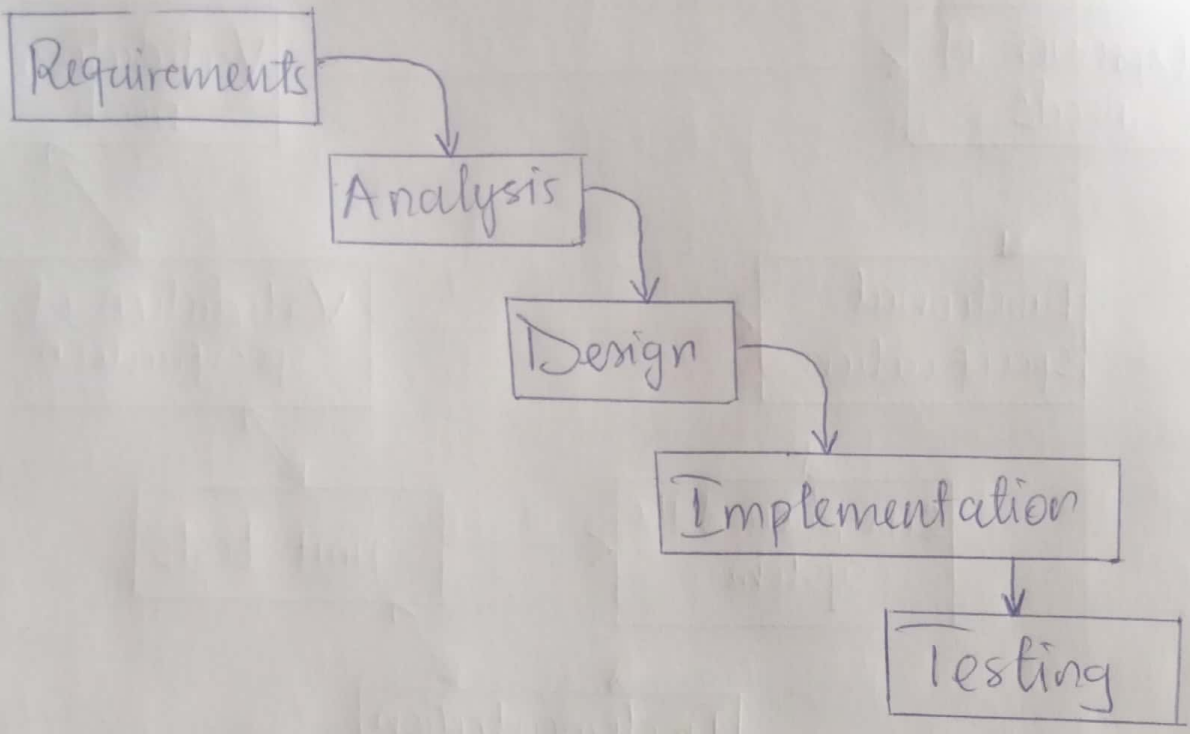
The V-model life cycle

The V-model proposes a cutting of the entire system, into smaller functional and technical sub-system. This modular management, makes easy team organisation. Reduces the complexity of analysis and design, makes verification and validation stages very easy.

The main inconvinience of this model resides in the late setting of implementation, tests and validation. Like in the sequential model, errors are very expensive because they can almost be detected almost at the end of the project.

3.4 The Waterfall Model.

The life-cycle of the waterfall model, respects the stages as represented below



The Waterfall Model

4.4.1 Requirements: In this stage, the requirements of the syste are clearly determined.

4.4.2 Analysis: The requirements are studied and classified and the overall architecture of the solution is determined. Eac major sub-system is analysed and its component classes are determined, also any interaction between compona are determined.

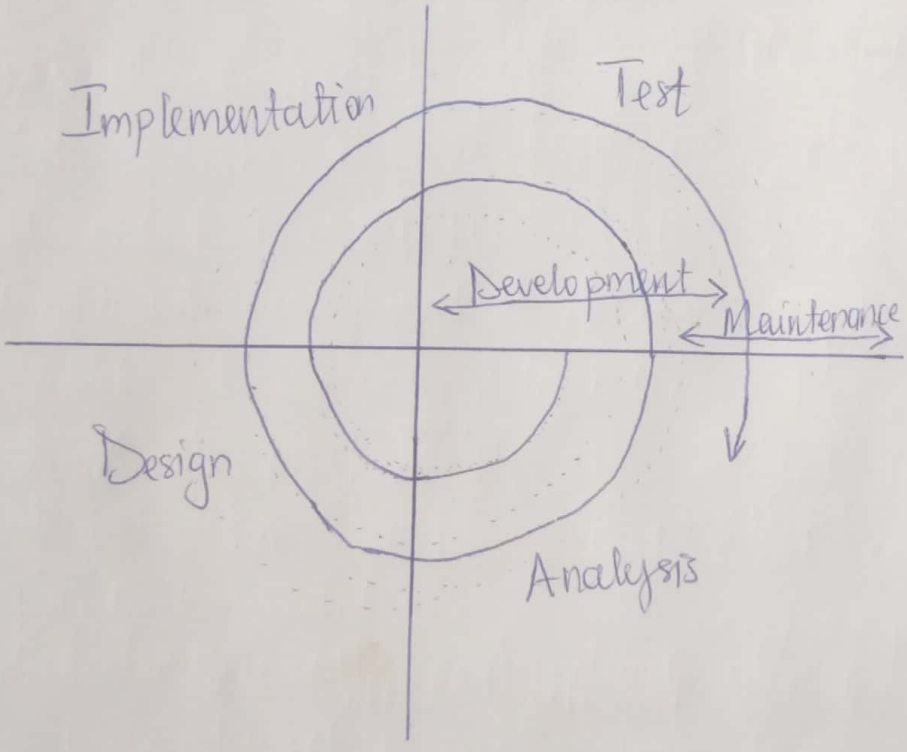
4.4.3 Design: Methods and data fields are designed for classes. Detailed algorithm for the methods are also defined

4.4.4 Implementation: Here, the individual classes and methods are coded in the target programming language.

4.4.5 Testing: The methods of each class, are tested in isolation (unit testing). The methods and classes are tested togethe (intergration testing) to verify that the work together and meet the requirements of the system.

Iterative Process or Spiral Model.

In such a process, all the stages are repeated as much as the validation is not satisfactory. It can be done in terms of superficial analysis, design, implementation and test of the whole application, then better analysis, design, implementation and tests, and improvements at each revolution. It can be seen in terms of thorough (detailed or deep) analysis, design, implementation and test of its parts and then analysis, design, implementation and test of a new parts at each revolution.



The Spiral Model

Model CA Questions.

1) Define software requirements
2) Give the characteristics of a good software
3. Elaborate on the metrics of a good software
4. List and explain the different software user ~~interfca~~ interface designs.
5. What are the different software requirement that you know
6. What is requirement engineering process and ~~is~~ it's stages?
7. Give requirements elicitation techniques.
8. Differentiate b/n SLC and SDLC
9. Give the d/f stages of SLC ~~and~~
10. Choose ~~and~~ a SDLC and explain it's stages.